A Meta-Learning Scheme for Adaptive Short-Term Network Traffic Prediction

Qing He*, Arash Moayyedi*, György Dán*, Georgios P. Koudouridis†, and Per Tengkvist†

*Division of Network and Systems Engineering, KTH Royal Institute of Technology, Stockholm, Sweden
Email: {qhe, moayyedi, gyuri}@kth.se
†RAN System Lab., Stockholm Research Center, Huawei Technologies, Stockholm, Sweden
Email: {george.koudouridis, per.tengkvist}@huawei.com

***Abstract*—Network traffic prediction is a fundamental prerequisite for dynamic resource provisioning in wireline and wireless networks, but is known to be challenging due to non-stationarity and due to its burstiness and self-similar nature. The prediction of network traffic at the user level is particularly challenging, because the traffic characteristics emerge from a complex interaction of user level and application protocol behavior. In this work we address the problem of predicting the network traffic at the user level over a short horizon, motivated by its applications in cellular scheduling. Motivated by recent works on robust adversarial learning, we treat the prediction problem for non-stationary traffic in an adversarial context, and propose a meta-learning scheme that consists of a set of predictors, each optimized to predict a particular kind of traffic, and of a master policy that is trained for choosing the best fit predictor dynamically based on recent prediction performance, using deep reinforcement learning. We evaluate the proposed meta-learning scheme on a variety of traffic traces consisting of video and non-video traffic. Our results show that it consistently outperforms state-of-the-art predictors, and can adapt to before unseen traffic without the need for retraining the individual predictors.**

***Keywords* – Meta learning, deep reinforcement learning, network traffic prediction, wireless networks.**

I. Introduction

Accurate network traffic prediction is becoming increasingly important for dynamic resource management in wireline and wireless networks. In wireline networks, for example, network traffic prediction can be used for reducing the power consumption in routers, for improving statistical multiplexing gains for quality of service provisioning, and for dynamic traffic routing in data centers [1]. In wireless networks, traffic prediction can be efficient in minimizing the power consumption of user equipment and of base stations through sleep scheduling, and through reduced complexity in traffic scheduling [2]–[5].

Depending on the application domain, the granularity of network traffic prediction can range from predicting the aggregate traffic intensity on a backbone link, i.e., the ratio of the time during which the link is occupied, often called the offered load, through predicting aggregates on local area networks, to predicting user level traffic characteristics. Similarly, the time scales for prediction can vary from day ahead prediction down to the millisecond level, depending on the intended use.

The main factors that affect traffic characteristics depend heavily on the level of aggregation and on the time scale. At the level of network aggregates self-similarity has long been observed, and sudden changes are rare due to statistical multiplexing across the traffic generated by many users and applications [5], [6]. Recent works have shown that feed-forward and recurrent neural networks can achieve fairly good prediction accuracy for such aggregate traffic [4], [5].

Prediction at the level of the traffic of individual users over short periods of time remains, however, extremely challenging due to sudden changes in traffic intensity inherent to various application protocols and due to the high impact of user behavior. For example, the most widely used protocol for video streaming, Dynamic Adaptive Streaming over HTTP (DASH) generates traffic with significant bursts, and its application behavior also depends on the user's media consumption behavior. At the same time, DASH streaming and live video are the largest contributors of network traffic [7]. Accurate prediction of such data traffic is thus of primary importance for increasing the efficiency of downlink scheduling in cellular networks and for sleep scheduling in user equipment.

In this work we address the problem of short term prediction of user traffic intensity, as a potential enabler of improved cellular scheduling. We first analyze state-of-the-art algorithms for time series prediction on a variety of data sets collected in a controlled environment, and show that they fail to provide accurate predictions across data sets. We then propose a meta-learning scheme that consists of a master policy and a number of sub-policies, predictors trained for particular types of traffic, motivated by recent works in robust adversarial machine learning [8], [9]. We evaluate the proposed meta-learning scheme on a variety of data sets containing a mix of video and non-video traffic, and our results show that it consistently outperforms all baseline schedulers.

The rest of the paper is organized as follows. In Section III we define the prediction problem. In Section IV we investigate traffic data and evaluate two commonly used predictors on homogeneous data sets. Based on the results we propose a meta-learning scheme and design the master and sub-policies in Section V. In Section VI we present extensive experimental results for the proposed scheme. Finally, we summarize the findings and discuss the next steps in Section VII.

II. Related Work

Network traffic prediction has received significant interest in the past three decades, both for traditional wired broadband networks and also for cellular radio access networks. In

general, methods from time series analysis have been used to model and predict network traffic. Generally, techniques for network traffic prediction can be organized in four broad categories, namely, linear time series model, nonlinear time series model, hybrid model, and decomposition-based model.

The most commonly used linear models are the autoregressive moving average (ARMA) model and the autoregressive integrated moving average (ARIMA) model [10], [11]. The ARMA model is a statistical model that can be used for predicting stationary time series data. In [12] and [13], ARMA has been employed to predict large file transfers and traffic patterns of the BitTorrent application, respectively. The ARIMA model applies ARMA to the time series after differencing, and can cope with trend-stationary time series [14]. The model order for ARIMA is usually obtained using the Box-Jenkins method [15]. Nonlinear time series are generated by non-linear dynamic equations. Nonlinear models such as the generalized auto regressive conditional heteroskedasticity (GARCH) model [16] and neural network techniques have become popular in recent years [17]. Notable amongst the non-linear models are approaches based on deep neural networks (DNN), which have been extensively used for short-term traffic prediction [5]. Compared to the linear models, DNNs provide significant improvements in prediction accuracy, as they contain numerous levels of non-linearities depending on the number of hidden layers, which allow them to efficiently represent highly nonlinear patterns and quickly-varying functional abstractions [18]. The experiments in [19] show superior performance of the prediction tool based on long short-term memory (LSTM) neural networks over the ARIMA models in cellular traffic classification and prediction. The combination of linear and nonlinear models is referred to as a hybrid model, which was found to give good results in the prediction and analysis of network traffic [20]. Decomposition based models decompose the time series into a trend component, a period component, a mutation component, and a random component, resulting in a decomposed model for predicting long term network traffic [21]. Originated in econometric analysis, this technique is not very efficient for short term prediction. An extensive review of previous work on network traffic prediction is given in [19], and shows that despite a variety of techniques, the rapidly changing nature of data traffic and its non-stationarity make it difficult and computationally expensive to train a one-size-fit-all predictor.

Our approach differs from existing techniques in that we do not aim at training a single predictor, but combine multiple predictors. In this respect our approach is close to recent works on ensemble learning, e.g., for road traffic prediction [22]. In particular, it is closest to the bucket of models ensemble technique with gating, but in our scheme the master policy chooses among the predictors in an online fashion. Going beyond existing techniques for ensemble learning, we adopt the meta-learning hierarchy (MLH) scheme [23], which has shown good promise in mitigating periods of adversarial attacks against reinforcement learning algorithms [8], [9]. In our proposed scheme a master policy chooses among a set of trained predictors depending on the characteristics of the traffic, and can learn in real-time based on the accuracy of the predictions.

III. Short Term Traffic Prediction Problem

We consider the problem of predicting the amount of IP traffic destined to a particular user in the network over the short term, with the objective of predicting the downlink traffic destined to a user equipment (UE) in a 4G cellular network, where there is a single data radio bearer (DRB) that carries all IP packets to a particular UE. Motivated by the periodicity of downlink scheduling, we consider that predictions are to be made periodically with a periodicity of τ; and are made for time periods that are integer multiples of τ; τ is typically 10 ms, 50 ms, or 100 ms. We refer to τ as the prediction interval.

The arriving traffic consists of a sequence of packets $i \geq 0$. We denote by t_i the arrival instant of packet i, and by b_i the size of packet i. Furthermore, we refer to $(t_i, b_i)_{i\geq 0}$ as the sequence of packet arrivals. Let us denote by $a(n)$ the number of bytes that arrive in the time interval $(n\tau, (n+1)\tau]$, i.e.,

$$a(n) = \sum_{i: n\tau < t_i \leq (n+1)\tau} b_i,$$

and we define $a(n, n') = \sum_{i=n}^{n'} a(n)$, i.e., the number of bytes that arrive in the time interval $(n\tau, (n'+1)\tau]$. Furthermore, we define the time series vector $A_m(n) = (a(n-m), \ldots, a(n-2), a(n-1))$, which contains the number of bytes arrived in the time interval $((n-m)\tau, n\tau]$, for some $m \leq n$. Clearly, vector A corresponds to the ground truth to be predicted.

To define the prediction problem, let us denote by Π_m the set of prediction models of order m. A prediction model $\pi_m \in \Pi_m$ is a real valued function that takes as input an m dimensional feature vector, i.e., $\pi_m : \mathbb{R}^m \to \mathbb{R}$. The m dimensional feature vector itself is the output of a mapping $H_m(t) \in \mathcal{H}_m$, which at time t maps the sequence $(t_i, b_i)_{t_i \leq t}$ to an m dimensional vector. Figure 1 shows the flow chart of prediction at time t.

To quantify the prediction error we use the mean square error (MSE), which for a predictor model $\pi_m \in \Pi_m$, mapping $H_m \in \mathcal{H}_m$ and a time period $(n\tau, n'\tau]$ can be expressed as

$$e_{\pi_m, H_m}(n\tau, n'\tau) = \frac{\sum_{\nu=n}^{n'} [a(\nu) - \pi_m(H_m(\nu\tau))]^2}{n' - n + 1}. \quad (1)$$

Note that in the above expression we make it explicit that the choice of the predictor π_m depends on feature engineering through H_m. Our objective is then to find a prediction model of at most order M that minimizes the expected MSE. Without loss of generality we can consider that the prediction problem

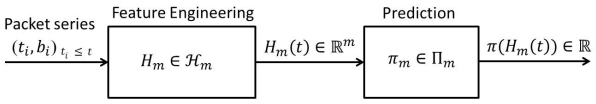


Fig. 1. Flow chart of traffic prediction: packet sequence is converted into a feature vector, which is fed into the predictor.

starts at time 0, and thus we are looking for a solution to the problem

$$\min_{\{\pi_m \in \Pi_m, H_m \in \mathcal{H}_m, 0<m\leq M\}} \quad \lim_{n\to\infty} e_{\pi_m, H_m}(0, n\tau). \tag{2a}$$

In the above formulation we explicitly distinguish between the feature engineering problem (i.e., finding H_m) and the problem of finding a good prediction model for a given set of features.

A common approach for solving the above problem is to consider that the mapping $H_m(n\tau) = A_m(n)$, i.e., the feature vector, is an m dimensional vector containing the number bytes that have arrived in the preceding m prediction intervals. In this case the above prediction problem becomes a standard time series forecasting problem, for which there are well established prediction models, e.g., Autoregressive Integrated Moving Average (ARIMA) and more recently, Long Short-term Memory (LSTM) neural networks. Existing models for time series forecasting either assume that the time series is stationary, or that it can be transformed into a stationary time series after compensating for seasonality and trend. These assumptions may be reasonable in economics, but network traffic is known to exhibit long range dependence and is non-stationary both when looking at a very large number of users, e.g., it shows the well known diurnal pattern [24], and when looking at individual users. It is thus important to develop computationally efficient prediction adaptive models that can cope with non-stationarity and with changes in traffic characteristics, which is the focus of our work.

IV. Predicting a Single Traffic Type

Before we present the proposed meta-learning scheme for network traffic prediction, in this section we show results for two widely used classes of predictors, ARIMA and LSTM. We will use the results in later sections for developing the proposed predictor based on a meta-learning scheme.

A. Data Sets

In order to obtain ground truth information, we captured network traffic using Wireshark on a desktop computer. We choose different kinds of network traffic based on the Internet traffic analysis reported in [7], and we captured four data sets of approximately 60 minutes each. The *Long DASH* data set was recorded while streaming a sequence of long video clips from YouTube using DASH. The *Short DASH* data set was recorded while streaming a sequence of short video clips from YouTube using DASH. The *Live video* data set was recorded while making a video call using Skype, and the *Non-video* data set was recorded while browsing news pages and other non-video content on the web using the browsers Chrome and Firefox. The data sets contain packet arrival time stamps and packet sizes, i.e., the sequence $(t_i, b_i)_{i\geq 0}$.

Considering the use case of cellular downlink scheduling, we set the prediction interval $\tau = 100$ ms, and we computed the number of arrived bytes per prediction interval, i.e., $H_m(n\tau) = A_m(n)$, where m is the order of the predictor. In the Appendix, we show the time series, the autocorrelation function (ACF), and the partial autocorrelation function (PACF) for the data sets, in Figures 12, 13, 14, and 15. We used the ACF and the PACF for analyzing the correlation structure of the time series [10].

B. Autoregressive Integrated Moving Average (ARIMA)

We start the evaluation with ARIMA models, which are widely used for time series analysis, including forecasting in econometrics and in engineering [10]. The formulation of the ARIMA(p,d,q) model is

$$\Delta^d P(t) = \delta + \phi_1 \Delta^d P(t-1) + \phi_2 \Delta^d P(t-2) + \cdots + \phi_p \Delta^d P(t-p) + A_t - \theta_1 A_{t-1} - \theta_2 A_{t-2} - \cdots - \theta_q A_{t-q}, \tag{3}$$

where $\Delta^d P(t)$ is the d^{th} differenced time series, ϕ_p and θ_q are coefficients of $P(t-p)$ and A_{t-q}, respectively. In addition, A_t is a (weak) white noise process, and

$$\delta = \left(1 - \sum_{i=1}^{p} \phi_i\right) \mu, \tag{4}$$

with μ denoting the process mean. If A_t follows the standardized student's t distribution, then

$$A_t = T_\mu \sqrt{\frac{\mu - 2}{\mu}}, \tag{5}$$

where T_μ is a Student's t distribution with $\mu > 2$ degrees of freedom.

Following common practice, we used the ACF and the PACF of the data sets to analyze the correlation structure of the time series and used the Box Jenkins method for identifying the best model order for each data set. We then used maximum likelihood estimation for finding the optimal model parameters, and we tested each of the four fitted models on all four data sets. Table I shows the 16 root mean square error (RMSE) values obtained using the ARIMA models. The first four rows of the table correspond to the fitted model for the Long DASH, Short DASH, Live video and Non-video data sets, respectively. The last row shows the average number of bytes that arrived per prediction interval, i.e., $\frac{1}{N}\sum_{n=1}^{N} a(n)$, and each column corresponds to a data set.

The results show that the best fit ARIMA model orders are different for the different data sets. Consistent with our expectation, we also observe that the lowest RMSE values are located on the main diagonal (not considering the last row), i.e., for each data set it is the model fitted to that data set that performs best. At the same time, a model fitted to a data set may result in a huge RMSE when used on a different data

TABLE I
RMSE results for 100 ms prediction interval, and average number of bytes per prediction interval.

Model	Long DASH	Short DASH	Live video	Non-video
LD.ARIMA(3,0,0)	25142	34607	65574	428
SD.ARIMA(6,0,1)	28356	26254	65542	689
LV.ARIMA(4,0,0)	59264	62331	18556	56261
NV.ARIMA(1,0,0)	27154	35280	65626	362
Average bytes	18403	4970	60330	385

set. As an extreme, the model fitted to the Live video data set results in an RMSE for Non-video data set two orders of magnitude higher than the average number of bytes per prediction interval.

Overall we observe that the lowest RMSE values are higher or approximately equal to the average number of bytes per prediction interval, which means that the prediction error is rather high. This is due to two main factors. On the one hand, three of the four time series exhibit large bursts of traffic that ARIMA cannot predict, as shown in Figures 12- 15 in the appendix. On the other hand, the time series are highly nonlinear. In addition, the Dickey-Fuller test, the KPSS test, the Phillips-Perron test and the variance ratio test all rejected the null hypothesis that any of the time series is a unit root process or that it is trend stationary.

C. Long Short-Term Memory (LSTM) Neural Network

As an alternative to ARIMA, we considered LSTM neural networks, which have recently proven to achieve good performance in a variety of prediction tasks. An LSTM is a recurrent neural network that is able to learn long-term dependencies in time series data [25]. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. The equations describing the operation of an LSTM are

$$i_t = \sigma_g\big(W_i x_t + U_i h_{t-1} + b_i\big) \tag{6a}$$
$$f_t = \sigma_g\big(W_f x_t + U_f h_{t-1} + b_f\big) \tag{6b}$$
$$o_t = \sigma_g\big(W_o x_t + U_o h_{t-1} + b_o\big) \tag{6c}$$
$$c_t = f_t * c_{t-1} + i_t * \sigma_c\big(W_c x_t + U_c h_{t-1} + b_c\big) \tag{6d}$$
$$h_t = o_t * \sigma_h(c_t) \tag{6e}$$

Here i, f, and o are the input, forget and output gates, respectively. Their outputs are described by similar equations with different parameter matrices $W \in \mathbb{R}^{h\times d}$ and $U \in \mathbb{R}^{h\times h}$, and bias vector parameters $b \in \mathbb{R}^h$, where the superscripts d and h refer to the number of input features and the number of hidden units, respectively. Vector $x_t \in \mathbb{R}^d$ is the input of the LSTM unit at current time t, and $c_t \in \mathbb{R}^h$ is the cell state vector. Vector $h_t \in \mathbb{R}^h$ is the output of the LSTM unit, computed by multiplying the memory with the output gate.

We used an LSTM network with an input layer, a hidden layer with one neuron, and an output layer that makes a single value prediction. We trained the LSTM for 1500 epochs (i.e., we trained it 1500 times on the data set) with a batch size of 1 (i.e., performed stochastic subgradient descent after every prediction step). We trained one LSTM model for each data set, based on the first 80% of the data, and we tested each of the four LSTM models on the remaining 20% of all four data sets that were not used for training. As the output given by a recurrent neural network may vary with different initial conditions, we repeated the experiments for each data set 10 times.

We show the average RMSE (excluding the extreme values) in Table II. The rows in the table correspond to the training data set, and hence the trained LSTM model, and each column corresponds to a test data set. The last row of the table shows the average number of arrived bytes per prediction interval, i.e., $\frac{1}{N}\sum_{n=1}^{N} a(n)$ (c.f. the last row of Table I). The results show that the LSTM model performs best for the data set that it was trained for, but the prediction error can be very high if an LSTM model is used on a data set with different type of traffic than what it was trained for.

TABLE II
LSTM results for 100 ms prediction interval.

Model	Long DASH	Short DASH	Live video	Non-video
LD.LSTM	619	23484	20812	507
SD.LSTM	2433	4829	17328	1505
LV.LSTM	9152	25781	16784	5313
NV.LSTM	7524	27641	44160	260
Average bytes	18403	4970	60330	385

In order to compare the prediction error of the ARIMA models and the LSTM models, in Figure 2 we show the RMSE normalized by the average number of bytes arrived per prediction interval for the two predictors. For each of the four data sets, when comparing the LSTM and the ARIMA model trained on this data set, we can observe that LSTM achieves a significantly lower prediction error than the corresponding ARIMA model. The improvement in prediction accuracy comes, however, at the cost of significant training time and increased prediction complexity.

Even more importantly, looking at the results for ARIMA (Table I) and for LSTM (Table II) we can also observe that only in very few cases a model trained on one data set performs well on other data sets; in general there is no one-size-fit-all predictor for data sets dominated by different types of traffic.

V. Meta-Learning for Traffic Prediction

Motivated by the results presented in Section IV, we are interested in developing a predictor that can dynamically adapt to highly varying and fast changing data traffic.

A. Feedback-based Predictor Architecture and Meta-learning

Based on the results presented in the previous section a possible solution would be a feed-forward predictor, which consists of a traffic classifier trained to recognize particular

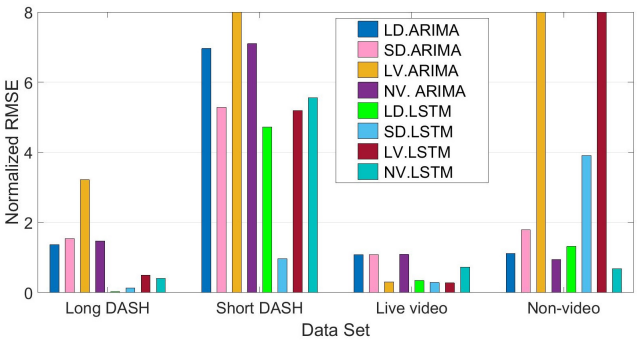
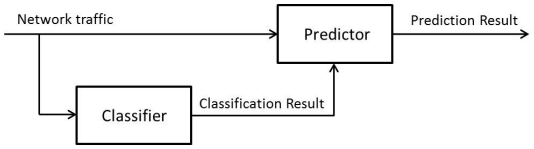
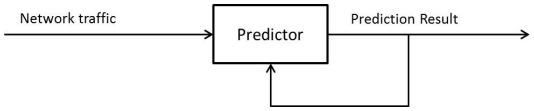


Fig. 2. Normalized RMSE for the ARIMA and the LSTM models with a prediction interval of 100 ms.



(a) Feed-forward architecture



(b) Feedback-based architecture

Fig. 3. Feed-forward and feedback-based architectures for traffic prediction. The feedback-based architecture does not need require a pre-trained classifier.

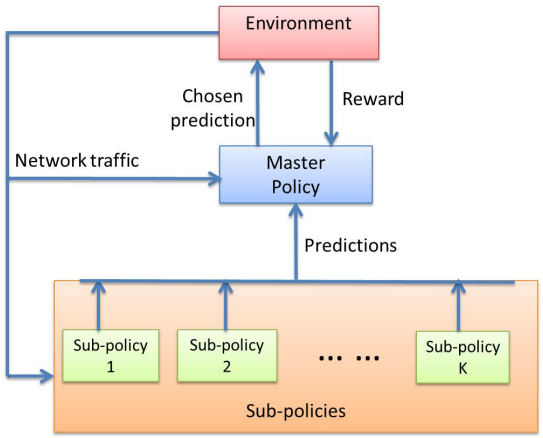


Fig. 4. The meta-learning scheme consists of a master policy and K sub-policies. The master policy chooses sub-policy based on past traffic and prediction performance, the sub-policies perform prediction.

kinds of traffic, e.g., live video, web traffic, file download, and a predictor that takes as input the historical data traffic and the result of the classification. This architecture, as illustrated in Figure 3a, is unfortunately rather impractical. First, it depends on the ability of training a traffic classifier, which requires ground truth about traffic classes (i.e., labelled data). Obtaining such ground truth is particularly difficult to an increasing share of encrypted traffic. Moreover, we argue that traffic classification for the purpose of short term traffic prediction may be not effective and actually counter-productive, because the traffic mix and the behavior of the different applications are continuously evolving, and thus one would require classifiers to be retrained periodically.

We thus propose an alternative, feedback-based architecture, where the predictor to be used is chosen based on observed prediction accuracy, not based on the notion of traffic classes. Figure 3b illustrates the architecture. The proposed architecture opens up for the possibility of dynamically managing a set of predictors, and could potentially be used for online traffic profiling as well. It allows to choose the best predictor for data traffic at any point in time, e.g., after a few initial prediction intervals), and allows for retraining the predictor(s) if the traffic characteristics change significantly without manual intervention. Moreover, it does not rely on a classifier trained for classifying different flows.

We propose to implement the feedback architecture based on a meta-learning scheme. Meta-learning schemes were recently proposed for making reinforcement learning robust to adversarial examples and have been considered for solving unknown tasks drawn from a known distribution in the machine learning literature [8], [9], [23]. The rationale for using a meta-learning scheme for the prediction of non-stationary traffic is based on its recent use in robust adversarial learning. There, a master policy chooses among a set of reinforcement learning agents; the individual agents are each trained for solving a particular kind of Markov decision process (MDP), some of which may be perturbed by an adversary. We can intuitively interpret non-stationary traffic as an adversary against individual predictors trained for predicting traffic with certain characteristics: the adversary presents traffic of different characteristics to the predictor, so as to increase the prediction error. The role of the master policy is then to choose a good predictor in response to such "adversarial" behavior.

The meta-learning scheme we propose consists of a master policy $\mu \in \mathcal{M}$ and a set $\mathcal{K}$, $K = |\mathcal{K}|$ of sub-policies. Each sub-policy $k \in \mathcal{K}$ is implemented by a predictor π^k trained using a certain kind of traffic. The master policy is responsible for deciding which sub-policy should be used for prediction during the next prediction interval. We show the proposed scheme in Figure 4. In what follows we discuss the proposed implementation of the master policy and of the sub-policies.

B. Master policy

As discussed above, at any point in time the master policy aims at choosing the best sub-policy for the next prediction interval. Considering that predictors such as LSTM have memory, the current choice of the sub-policy may affect the future accuracy of the sub-policies. Thus the objective of the master policy is to find a policy $\mu^*_{m'} \in \mathcal{M}$ that minimizes the expected average MSE, i.e.,

$$\min_{\mu_{m'} \in \mathcal{M}} \lim_{n \to \infty} \frac{\sum_{\nu=0}^{n} [a(\nu) - \mu_{m'}(H'_{m'}(\nu\tau))\{H_m(\nu\tau)\}]^2}{n+1}, \tag{7}$$

where without loss of generality we denote the decision time instant by 0. Note that $\mu_{m'} : \mathbb{R}^{m'} \to \mathcal{K}$, i.e., the master policy is based on features $H'_{m'}(\nu\tau)$ of the past arriving traffic for some $m' \geq 0$ and chooses a sub-policy π^k_m, which itself performs prediction, based on the feature mapping $H_m(\nu\tau)$. In general, m' used by the master policy need not be the same as the predictor order m used for the sub-policies.

Observe that the problem of the master policy can be formulated as an MDP $(\mathcal{S}, \mathcal{K}, P_k(s,s'), R_k(s,s'))$. For a given m' the state space of the MDP is $\mathcal{S} = \mathbb{Z}^{m'}_{\geq 0}$, the action set is $\mathcal{K}$, and the immediate reward when choosing action k (i.e., sub-policy π^k_m) and transitioning from state $A_{m'}(n)$ to $A_{m'}(n+1)$ is

$$R_k(A_{m'}(n), A_{m'}(n+1)) = \left[a(n) - \pi^k_m(H'_{m'}(n\tau))\right]^2,$$

i.e., the squared prediction error. In what follows, we refer to the value of m' as the *dimension of the state space* of the master policy. Importantly, the state transition probabilities $P_k(s,s')$ are not known, as they depend on the type of traffic.

Training the master policy: In the case of short-term traffic prediction, the state transition probabilities are unknown. We, therefore, propose to use (deep) reinforcement learning for learning the master policy. The master policy chooses a predictor and evaluates the resulting reward for every prediction interval. Observe that since prediction is done in real time upon every prediction interval, and the reward (squared prediction error) is directly observable, learning can be performed in real-time. This allows one to train the master policy on actual network traffic, and to retrain the master policy on-demand.

C. Sub-policies

In the meta-learning scheme we propose that each sub-policy $k \in \mathcal{K}$ is a predictor π^k trained for a particular kind of traffic or traffic mix. The sub-policies may be pre-trained, as in the performance evaluation presented in the following section. Nonetheless, the proposed scheme also allows for a predictor to be trained in real-time - together with the master policy - based on the actual traffic and the prediction results it provides. We leave this interesting direction to be subject of future work, and rely on pre-trained predictors as sub-policies in the following evaluation.

VI. Performance Evaluation

In what follows we evaluate the proposed meta-learning scheme based on a variety of traffic traces. We first describe the sub-policies used, then the evaluation methodology, and finally we provide numerical results.

A. Sub-policies

We used seven predictors as sub-policies for the evaluation of the proposed meta-learning scheme. The first four predictors are the LSTM predictor trained for the data sets with a single type of traffic, namely, the long DASH predictor, the short DASH predictor, the live video predictor, and the non-video predictor. Considering that these predictors may not perform well for a variety of traffic, we introduced three additional predictors.

Persistence model forecast (PMF): PMF assumes the time series is persistent, and hence it uses the last observed value as the prediction. That is, the predictor is given by

$$\pi^{PMF}(n+1) = a(n). \tag{8}$$

The PMF predictor is expected to be used when the number of arriving bytes is approximately constant.

Zero predictor (ZP): ZP outputs a constant value of zero, and is expected to be used during idle periods.

Average value predictor (AVP): AVP outputs the average number of bytes that arrived during the last m prediction periods. Using a recursive formulation it is given by

$$\pi_m^{AVP}(n+1) = \pi_m^{AVP}(n) + \frac{a(n) - a(n-m-1)}{m}. \tag{9}$$

The above three predictors are all computationally inexpensive. Figure 5 shows the complete learning scheme.

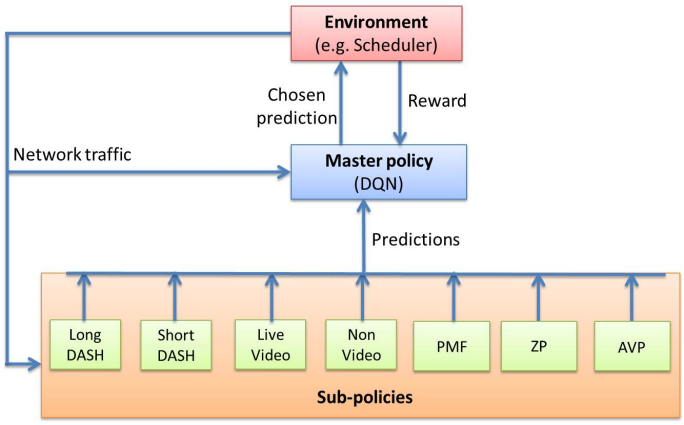


Fig. 5. The proposed meta-learning scheme including the 7 sub-policies considered in the evaluation.

B. Evaluation methodology

Data sets: We used in total 12 data sets containing incoming traffic on a desktop computer captured using Wireshark. The first four data sets are the ones used in Section IV, and are dominated by a single type of traffic, namely long DASH, short DASH, live video, and non-video data (Data sets 1-4). We refer to these four data sets as single-type data sets. Another four traces contain traffic traces collected during daily work and during leisure time, and hence contain a mix of various kinds of traffic. We refer to these four as mixed data sets (Data sets 5-8). In general, each data set spans over tens of thousands of prediction intervals. In addition we evaluated the predictors on four publicly available data sets reported in [19]. In the appendix, Table IV provides a summary of the data sets.

Master policy: We trained a Deep Q-network (DQN) agent for the master policy using stochastic gradient descent. A DQN consists of an input layer, an output layer, and multiple hidden layers. Each layer has a number of neurons. The number of hidden layers, number of neurons per layer, as well as parameters such as learning rate and the number of learning steps affect the trained model and the prediction performance. In the following we refer to the number of hidden layers and the number of neurons per layer as the neural network structure. The learning rate influences the convergence of learning through controlling the weight of the update after each batch in stochastic gradient descent. Finally, the number of steps is the number of samples used for training and is typically in the order of millions, achieved through training on a particular data set over multiple epochs. We provide the network structure, learning rate and the number of steps in the tables provided in the appendix. As input features for the DQN we used the number of bytes that arrived in the past m' prediction intervals, i.e., $H'_{m'}(n\tau) = A_{m'}(n)$.

Implementation: We implemented the proposed meta-learning scheme in Python. We developed LSTM networks using the Keras deep learning library to be employed as sub-policies. For the master policy we used Keras deep reinforcement learning libraries to create a DQN agent that interacts with the environment, which is implemented as a custom scenario in OpenAI Gym.

Performance metric: We use the RMSE as the performance metric to evaluate the predictors.

Oracle policy: As a baseline, for each data set and the set of predictors we calculate a lower bound, which we obtain using a hypothetical oracle policy that in every prediction interval takes the best action, i.e., selects the predictor that provides the most accurate prediction of the arrived bytes. The meta-learning scheme with the same set of predictors can clearly not perform better than this oracle, hence it gives the lower bound of the achievable RMSE.

EXP3.S: As an additional basis for comparison we used EXP3.S, which is an online algorithm for the multi-armed bandit problem that at every prediction interval aims at finding the best expert based on past performance of the experts [26]. For EXP3.S we tune the parameter settings according to Corollary 8.2 in [26] for the forgetting factor α and the exploration factor γ, and we set T to be equal the length of the data set.

Hardware platform: We performed all evaluations on an NVIDIA Quadro M2000 GPU with 768 cores and 4GB of memory, in a HP Proliant ML110 Gen9 server with Intel Xeon E5-2620 CPU and 32 GB RAM.

C. Numerical results

Single type data set: We start the evaluation using the meta-learning scheme with four sub-policies, namely the Long DASH, the short DASH, the live video, and the non-video predictors, thus $|\mathcal{K}| = 4$. We first performed a test on a subset of Data set 3 containing 5000 samples collected during a video call, thus live video data dominates the data set. For the master policy we trained a DQN that has three densely connected hidden layers, each with 16 neurons. The input layer of the DQN has $m' = 2$ two neurons, i.e., the input to the DQN is $A_2(n)$. Table III shows the RMSE values obtained when using one of the four predictors only, as well as using the meta-learning scheme. The results show that the proposed meta-learning scheme works very well, as it significantly outperforms any single predictor. This observation implies that for some time intervals, the predictor trained on this data set may perform worse than other predictors. The reason for this phenomenon is that even the data sets dominated by a single type of traffic contain a certain amount of other types of traffic. For example, watching short Youtube videos inherently involves downloading the webpage through which the videos are available. The predictor trained for short video traffic performs well overall for the data set of short videos, but the meta-learning scheme can recognize the periods when traffic is actually not generated by watching video traffic.

TABLE III
Prediction results for Data set 3.

Data set 0: 5000 samples, single-type data set, data type: video call, prediction interval: 100ms.					
Predictor	Long DASH	Short DASH	Live video	Non-video	Meta-learning
RMSE	27696	20468	18056	50797	15023

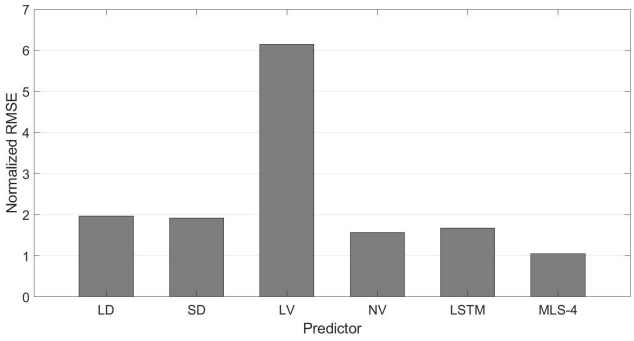


Fig. 6. Prediction results for Data set 5: 50000 samples, mixed traffic, prediction interval $\tau = 100$ms.

Mixed data set: Next, we evaluated the predictors on mixed data sets, starting with Data set 5. Table V in the Appendix shows information about the data set and the RMSE results when using the four predictors trained for a single type of traffic, together with the oracle policy. Besides the four predictors trained for a single type of traffic and the oracle policy, the table also shows the prediction error for an LSTM predictor trained for this mixed data set. Figure 6 shows the normalized RMSE for the five simple predictors and the meta-learning scheme with $K = 4$ sub-policies, i.e., the predictors trained for a single type of traffic. The figure shows that the meta-learning scheme achieves significantly lower RMSE than any of the individual predictors, including the LSTM. Furthermore, it is worth noting that the gap between the best result and the oracle policy is less than 5%. Table VI in the Appendix shows all results obtained using various DQNs as the master policy in the proposed meta-learning scheme, and shows that the proposed meta-learning outperformed the individual predictors using almost all considered DQNs (except one).

Next we evaluated all seven predictors on a large mixed data set with 135000 samples, referred to as Data set 6. The RMSE results of the individual predictors are shown in Table VII in the Appendix. Figure 7 shows the RMSE results of the 7 individual predictors described in Section VI-A, an LSTM predictor trained on Data set 8, the RMSE obtained by the meta-learning scheme with the 4 single type predictors as sub-policies (MLS-4), by the meta-learning scheme with all 7 sub-policies (MLS-7) , and by EXP3.S. All the RMSE values are normalized by the RMSE of the oracle policy using the 7 sub-policies. The figure shows that even when using 4 sub-policies, the proposed meta-learning scheme achieves lower RMSE, and the performance of the meta-learning scheme is further improved by considering 7 sub-policies (Tables VIII and IX in the Appendix show the DQN parameters and numerical results). Comparing the results with 4 and 7 sub-policies we can conclude that adding more predictors as sub-policies can significantly improve the performance of the meta-learning scheme. In particular, the RMSE of the best result with 7 predictors is 48% lower compared to that with 4 predictors. The meta-learning scheme also outperforms EXP3.S. This is reasonable considering that it takes a significant amount of time for EXP3.S to adapt its choice as the traffic changes, even with the considered parameters, which were chosen to

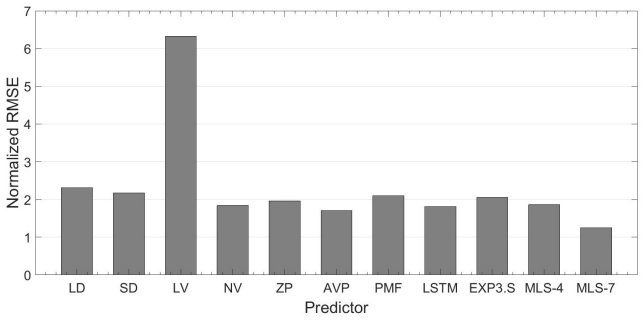


Fig. 7. Prediction results for Data set 6: 135000 samples, mixed traffic, prediction interval $\tau = 100$ms.

match the traces.

Finally, we further evaluate the meta-learning scheme with seven predictors by testing it on two data sets consisting of mixed traffic. Figure 8 shows the normalized RMSE for all predictors, the meta-learning scheme (Tables X - XIII in the Appendix contain the numerical values), and the EXP3.S algorithm. The results show that the meta-learning scheme provides a very accurate prediction for both data sets, only 17% and 9% above the error of the respective oracle policy, and it consistently outperforms EXP3.S as well.

Publicly available data set: We also evaluate the meta-learning scheme on four data sets created from the Wireshark traces collected and analyzed in [19], available on GitHub. The data set contains cellular uplink and downlink traffic, of which we only considered the downlink traffic. We extracted 4 data sets of 1 hour duration each, which we refer to as Data sets 9 to 12. Data sets 9 and 10 correspond to traffic during working hours on a weekday, while the other two data sets consist of off-peak traffic collected in the early morning of a weekend day. It is important to note that we are not aware of the traffic types present in the traces, we only extracted the packet arrival times and packet sizes. We then used our 7 predictors as sub-policies, of which four were trained on the data sets we collected, and trained the master policy on Data set 9.

Figure 9 shows the normalized RMSE of the 7 predictors used as sub-policies, of an LSTM predictor trained on the

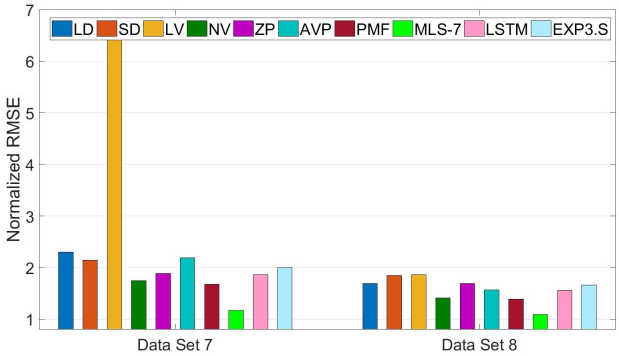


Fig. 8. Prediction results for Data set 7: 85000 samples, mixed traffic, and for Data set 8: 80000 samples, mixed traffic. Both with prediction interval $\tau = 100$ms.

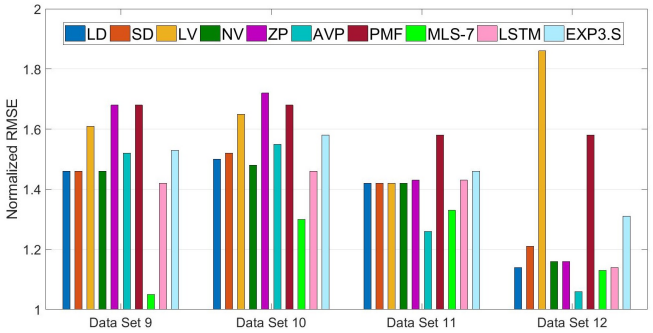


Fig. 9. Prediction results for Data sets 9, 10, 11, and 12. All data sets contain 30000 samples, mixed traffic, and have a prediction interval of $\tau = 100$ms.

mixed data set, of the meta-learning scheme with 7 sub-policies for the four data sets, and of the EXP3.S algorithm. The results show that the meta-learning scheme performs best among all predictors for Data sets 9 and 10, i.e., for the data sets with significant traffic. For the two data sets with little traffic the average value predictor performs best, which indicates that the traffic is periodic and has low intensity (consistent with the RMSE values in Tables XVII and XVIII in the Appendix).

D. Computational cost

Considering the intended real-time operation of the proposed meta-learning scheme, we now assess the computational cost of the predictor. The proposed meta-learning scheme consists of a DQN for the master policy and a number of LSTM networks as sub-polices. If the DQN and the LSTMs can be executed in parallel then the inference time is determined by the maximum of the inference times of the DQN and the LSTMs. As we use relatively small LSTM networks, it is likely that the inference time of the DQN is dominant. The inference time of the DQN itself is determined by the network structure; for a DQN with m layers, each containing at most n neurons, the complexity is bounded by $O(n^m)$. The DQNs used in our implementation contain relatively few layers (as shown in the appendix), thus the computational cost of inference is very low.

Figure 10 shows the minimum, average, and maximum inference times of 9 implementations of MLS-4 and MLS-7. The implementations differ in terms of the deep neural network used as the master policy. The results show that with the hardware platform described in Section VI-B the inference takes on average 1 millisecond. The results indicate that the number of neurons in the hidden layer has little impact on the inference time (due to parallel processing) but the dimension of the state and the number of layers do affect it. The former we attribute to the need for copying more data from the GPU memory to the cores, the latter is a natural consequence of the sequential execution across layers.

E. Discussion

Overall the proposed meta-learning scheme works very well, e.g., for most data sets it performed very close to the

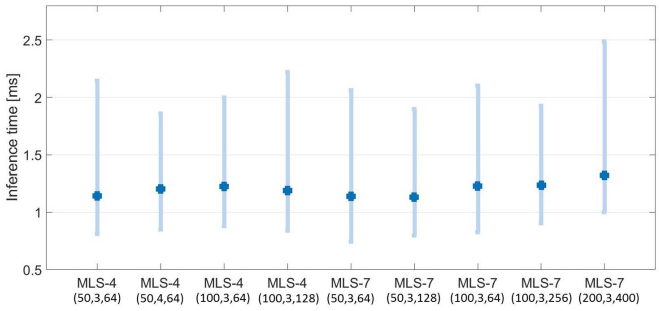


Fig. 10. Inference time of DQN(m, b, c) in MLS-4 and MLS-7, where m is the dimension of state, b is the number of hidden layers, and c the number of neurons in each layer.

oracle policy. Overall, based on the results of our evaluation we make the following observations.

- For a given DQN structure and training parameters, increasing the dimension of the state can improve the results, implying that the historical information could help the learning model (c.f., Table IX).
- Comparing results for different data sets, the optimal choice of DQN parameters is influenced by the data set size. This can be attributed to the fact that a longer data set would exhibit more diverse traffic patterns, and thus a network with more hidden layers and neurons may be needed. The diversity of traffic patterns needs thus to be taken into account.
- Adding more predictors may have a high potential for improving the prediction results (c.f. Figure 7). Moreover, the computation cost at inference time would not grow too much if the additional predictors are simple (e.g., a constant predictor). For the master policy, the training cost increases slowly with the number of sub-policies.
- Based on the results obtained on the public data sets we can conclude that the meta-learning scheme is able to generalize rather well, as it achieved low prediction error without retraining the predictors used as sub-policies. More importantly, the performance of the meta-learning scheme can be further improved by improving the prediction performance of the sub-policies, and as such it can provide a versatile framework for traffic prediction.

We envision a variety of application domains for short term prediction based on the proposed meta-learning scheme. First, short term traffic prediction could be used for semi-persistent downlink scheduling, with the objective of reducing the scheduling complexity. In semi-persistent scheduling, the mobile terminals to be scheduled can be chosen based on the amount of traffic they are predicted to receive in the subsequent scheduling intervals, and scheduling decisions can be made for multiple subsequent scheduling intervals. It is so far unclear whether such semi-persistent scheduling could be achieved without significant impact on fairness and throughput. Figure 11 shows the flow-chart of a potential architecture, where the dashed line shows that in principle the master policy could also be trained based on the resulting scheduling performance. Second, short term downlink traffic prediction could be used for base station sleep scheduling [27], with

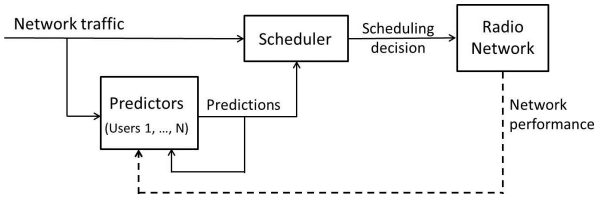


Fig. 11. Block diagram of scheduling based on traffic prediction. Dashed line indicates that the master policy could be trained based on the scheduling performance as well.

the objective of reducing energy consumption. Using traffic prediction could make it possible to optimize sleep scheduling subject to constraint on the introduced latency, based on the predicted traffic arrival. Exploring these applications of short term traffic prediction is subject of our ongoing work.

VII. Conclusion and Outlook

We considered the problem of user level traffic prediction over a short time horizon, motivated by its potential use in cellular resource management. We proposed a meta-learning scheme that dynamically selects a predictor from a set of predictors, and thereby allows to adapt to changing traffic characteristics. Our performance evaluation based on collected and public traffic traces shows that the proposed meta-learning scheme outperforms standalone predictors significantly, and could be an efficient solution for the short term prediction of network traffic despite changing traffic characteristics.

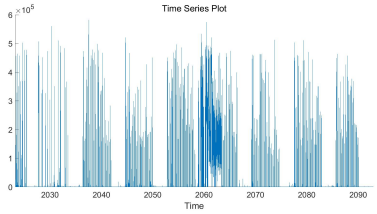
A very promising, but so far unexplored feature of the considered meta-learning scheme is that it enables updating the set of predictors in real-time, providing the ability to adapt to changes in traffic patterns over long time scales. Adding a new predictor involves retraining the master policy, and thus exploring the technical feasibility of this approach, in terms of computational cost and the level of adaptation it allows, could be an interesting avenue of future research.

Appendix

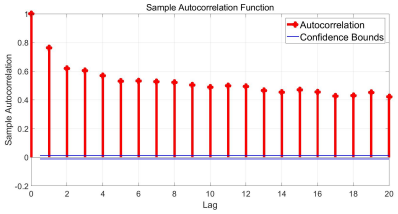
In the appendix we provide supporting data for the results presented in the paper. We first provide a summary of the data sets in Table IV, then we show the time series, the ACF and the PACF of the four single type data sets used in the evaluation. After that we provide the RMSE values obtained by the predictors and the meta-learning scheme for the different data sets in tabular format.

TABLE IV
Summary of Data sets.

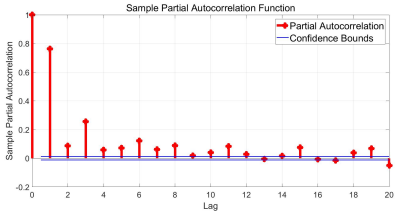
Data set index	Data set size	No. of traffic classes	Types of traffic classes	Properties
1	30000	Single	Long video	Private
2	30000	Single	Short video	Private
3	30000	Single	Live video	Private
4	30000	Single	Non-video	Private
5	50000	Multiple	Mixed traffic	Private
6	135000	Multiple	Mixed traffic	Private
7	85000	Multiple	Mixed traffic	Private
8	80000	Multiple	Mixed traffic	Private
9 - 12	30000	Multiple	Mixed traffic	Public



(a) Time Series

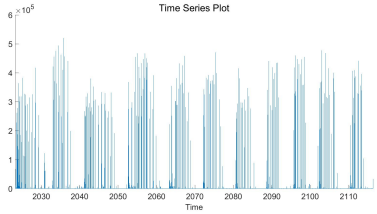


(b) Autocorrelation function (ACF),

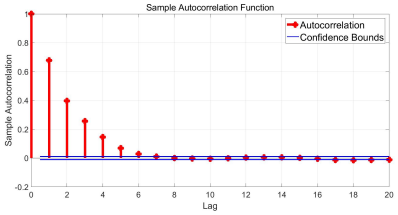


(c) Partial autocorrelation function (PACF)

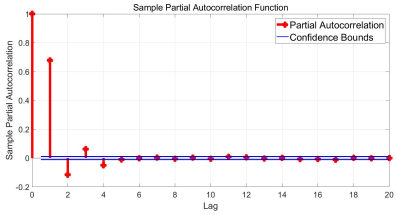
Fig. 12. Time series, ACF and PACF of Long DASH data set with $\tau = 100ms$.



(a) Time Series

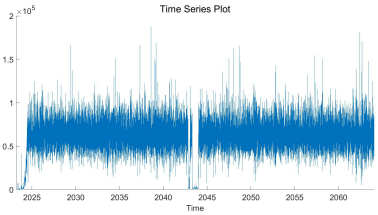


(b) Autocorrelation function (ACF)

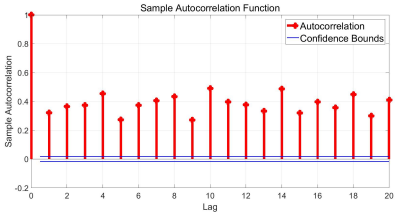


(c) Partial autocorrelation function (PACF)

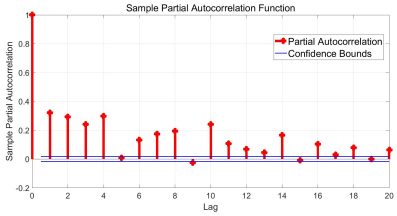
Fig. 13. Time series, ACF and PACF of Short DASH data set with $\tau = 100ms$.



(a) Time Series

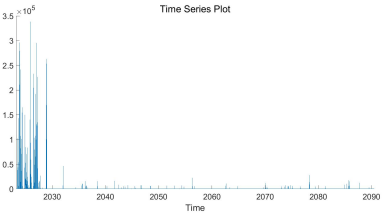


(b) Autocorrelation function (ACF)

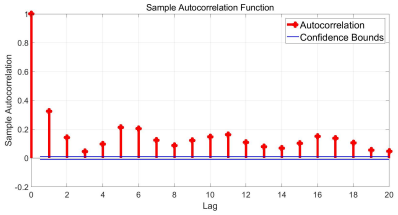


(c) Partial autocorrelation function (PACF)

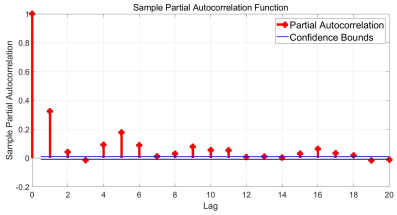
Fig. 14. Time series, ACF and PACF of Video call data set with $\tau = 100ms$.



(a) Time Series



(b) Autocorrelation function (ACF)



(c) Partial autocorrelation function (PACF)

Fig. 15. Time series, ACF and PACF of Non-video data set with $\tau = 100ms$.

TABLE V
RMSE results for Data set 5 with the four predictors used as sub-policies and using an LSTM trained on this data set.

Data set 5: 50000 samples, mixed data set, prediction interval: 100ms.						
Predictor	Long DASH	Short DASH	Live video	Non-video	Oracle	LSTM
RMSE	144k	141k	450k	115k	73k	122k
Normalized RMSE	1.97	1.92	6.15	1.57	1	1.67

TABLE VI
DQN parameters and the resulting RMSE when using the meta-learning scheme with four sub-policies for Data set 5.

Data set 5: 50000 samples, mixed data set, prediction interval: 100ms.					
Dimension of state	NN structure	Memory limit	Learning rate	Number of steps	Normalized RMSE
10	64+64+64	30000	0.0001	22M	1.51
50	64+64+64	30000	0.0001	25M	1.39
10	64+64+64	30000	0.00001	25M	1.62
10	64+64+64	300000	0.0001	25M	1.47
50	64+64+64+64	500000	0.0001	25M	1.47
100	128+128+128	500000	0.0001	25M	1.05
100	64+64+64	500000	0.0003	25M	1.28

TABLE VII
RMSE results for Data set 6 with the seven predictors used as sub-policies, an LSTM trained on mixed data, and EXP3.S.

Data set 6: 135000 samples, mixed data set, prediction interval: 100ms.										
Predictor	Long DASH	Short DASH	Live video	Non-video	Zero	Average	PMF	Oracle	LSTM	EXP3.S
RMSE	101030	94920	277510	80690	85850	74320	92001	43821	79144	90206
Normalized RMSE	2.31	2.17	6.33	1.84	1.96	1.70	2.10	1	1.81	2.06

TABLE VIII
DQN parameters and the resulting RMSE when using the meta-learning scheme with four sub-policies for Data set 6.

Data set 6: 135000 samples, mixed data set, prediction interval: 100ms.						
Dimension of state	NN structure	Memory limit	Learning rate	Number of Steps	RMSE	Normalized RMSE
100	128+128+128	500000	0.0001	67500000	81356	1.86
50	64+64+64	500000	0.0001	67500000	84277	1.92
100	64+64+64	500000	0.0003	67500000	93618	2.14

TABLE IX
DQN parameters and the resulting RMSE when using the meta-learning scheme with seven sub-policies for Data set 6.

Data set 6: 135000 samples, mixed data set, prediction interval: 100ms.						
Dimension of state	NN structure	Memory limit	Learning rate	Number of Steps	RMSE	Normalized RMSE
50	128+128+128	500000	0.0001	20000000	72250	1.65
100	256+256+256	500000	0.0001	20000000	60600	1.38
200	300+300+300	500000	0.0003	20000000	54978	1.25
100	64+64+64	500000	0.0001	20000000	75423	1.72
50	64+64+64	500000	0.0001	20000000	92795	2.12
100	300+300+300	500000	0.0001	20000000	59640	1.36

TABLE X
RMSE results for Data set 7 with the seven predictors used as sub-policies, an LSTM trained on mixed data, and EXP3.S.

Data set 7: 85000 samples, mixed data set, prediction interval: 100ms.										
Predictor	Long DASH	Short DASH	Live video	Non-video	Zero	Average	PMF	Oracle	LSTM	EXP3.S
RMSE	118520	110170	346220	90220	97470	112700	86370	51485	95780	103409
Normalized RMSE	2.30	2.14	6.72	1.75	1.89	2.19	1.68	1	1.86	2.00

TABLE XI
DQN parameters and the resulting RMSE when using the meta-learning scheme with seven sub-policies for Data set 7.

Data set 7: 85000 samples, mixed data set, prediction interval: 100ms.						
Dimension of state	NN structure	Memory limit	Learning rate	Number of Steps	RMSE	Normalized RMSE
200	400+400+400	1000000	0.0001	20000000	60365	1.17

TABLE XII
RMSE results for Data set 8 with the seven predictors used as sub-policies, an LSTM trained on mixed data, and EXP3.S.

Data set 8: 80000 samples, mixed data set, prediction interval: 100ms.										
Predictor	long DASH	short DASH	live video	non-video	Zero	Average	PMF	Oracle	LSTM	EXP3.S
RMSE	76805	83647	84301	63887	76610	71300	62926	45315	70907	75396
Normalized RMSE	1.69	1.85	1.86	1.41	1.69	1.57	1.39	1	1.56	1.66

TABLE XIII
DQN parameters and the resulting RMSE when using the meta-learning scheme with seven sub-policies for Data set 8.

Data set 8: 80000 samples, mixed data set, prediction interval: 100ms.						
Dimension of state	NN structure	Memory limit	Learning rate	Number of Steps	RMSE	Normalized RMSE
300	400+400+400	1000000	0.0001	20000000	49544	1.09

TABLE XIV
RMSE results for Data set 9 with the seven predictors used as sub-policies, an LSTM trained on a mixed data set, and EXP3.S.

Data set 9: 30000 samples, mixed data set, prediction interval: 100ms.										
Predictor	Long DASH	Short DASH	Live video	Non-video	Zero	Average	PMF	Oracle	LSTM	EXP3.S
RMSE	24596	24589	27080	24694	28355	25632	28281	16866	23876	25922
Normalized RMSE	1.46	1.46	1.61	1.46	1.68	1.52	1.68	1	1.42	1.53

TABLE XV
DQN parameters and the resulting RMSE when using the meta-learning scheme with seven sub-policies for Data set 9

Data set 9: 30000 samples, mixed data set, prediction interval: 100ms.						
Dimension of state	NN structure	Memory limit	Learning rate	Number of Steps	RMSE	Normalized RMSE
200	400+400+400	1000000	0.0001	20000000	17659	1.05

TABLE XVI
RMSE for Data set 10 using the seven predictors, an LSTM, the meta-learning scheme with seven sub-policies, and EXP3.S.

Data set 10: 30000 samples, mixed data set, prediction interval: 100ms.											
Predictor	Long DASH	Short DASH	Live video	Non-video	Zero	Average	PMF	Oracle	MLS-7	LSTM	EXP3.S
RMSE	24027	24375	26545	23801	27558	24899	26986	16042	20854	23365	25350
Normalized RMSE	1.50	1.52	1.65	1.48	1.72	1.55	1.68	1	1.30	1.46	1.58

TABLE XVII
RMSE for Data set 11 using the seven predictors, an LSTM, the meta-learning scheme with seven sub-policies, and EXP3.S.

Data set 11: 30000 samples, mixed data set, prediction interval: 100ms.											
Predictor	Long DASH	Short DASH	Live video	Non-video	Zero	Average	PMF	Oracle	MLS-7	LSTM	EXP3.S
RMSE	3176	3177	3177	3183	3187	2809	3531	2235	2974	3185	3259
Normalized RMSE	1.42	1.42	1.42	1.42	1.43	1.26	1.58	1	1.33	1.43	1.46

TABLE XVIII
RMSE for Data set 12 using the seven predictors, an LSTM, the meta-learning scheme with seven sub-policies, and EXP3.S.

Data set 12: 30000 samples, mixed data set, prediction interval: 100ms.											
Predictor	Long DASH	Short DASH	Live video	Non-video	Zero	Average	PMF	Oracle	MLS-7	LSTM	EXP3.S
RMSE	2648	2805	4311	2698	2697	2458	3658	2320	2628	2635	3042
Normalized RMSE	1.14	1.21	1.86	1.16	1.16	1.06	1.58	1	1.13	1.14	1.31

REFERENCES

[1] A. Cenedese, F. Tramarin, and S. Vitturi, "An energy efficient ethernet strategy based on traffic prediction and shaping," *IEEE Transactions on Communications*, vol. 65, no. 1, pp. 270–282, Jan 2017.

[2] U. Paul, M. M. Buddhikot, and S. R. Das, "Opportunistic traffic scheduling in cellular data networks," in *2012 IEEE International Symposium on Dynamic Spectrum Access Networks*, Oct 2012, pp. 339–348.

[3] R. Li, Z. Zhao, J. Zheng, C. Mei, Y. Cai, and H. Zhang, "The learning and prediction of application-level traffic data in cellular networks," *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3899–3912, June 2017.

[4] H. D. Trinh, L. Giupponi, and P. Dini, "Mobile traffic prediction from raw data using lstm networks," in *IEEE Intl. Symp. on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2018, pp. 1827–1832.

[5] C. Huang, C. Chiang, and Q. Li, "A study of deep learning networks on mobile traffic forecasting," in *IEEE Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, Oct 2017, pp. 1–6.

[6] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of ethernet traffic (extended version)," *IEEE/ACM Transactions on Networking*, vol. 2, no. 1, pp. 1–15, Feb 1994.

[7] Cisco Systems, "Cisco visual networking index: Forecast and trends, 2017-2022 White Paper," Tech. Rep., updated in 2019, https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf.

[8] X. Lee, A. Havens, G. Chowdhary, and S. Sarkar, "Learning to cope with adversarial attacks," in *Proc. of Intl. Conf on Machine Learning (ICML)*, 2019.

[9] A. J. Havens, Z. Jiang, and S. Sarkar, "Online robust policy learning in the presence of unknown adversaries," in *Proc. of Conference on Neural Information Processing Systems (NIPS)*, 2018.

[10] T. Anderson, "The statistical analysis of time series," *J. Wiley and Sons*, 1971.

[11] H. Feng and Y. Shu, "Study on network traffic prediction techniques," in *Proc. of International Joint Conference on Wireless Communications Networking and Mobile Computing*, 2005, pp. 1041–1044.

[12] N. K. Hoong, P. K. Hoong, I. K. T. Tan, N. Muthuvelu, and L. C. Seng, "Impact of utilizing forecasted network traffic for data transfers," in *13th International Conference on Advanced Communication Technology (ICACT2011)*, 2011, pp. 1199–1204.

[13] P. K. Hoong, I. K. T. Tan, and C. Y. Keong, "BitTorrent network traffic forecasting with ARIMA," *International Journal of Computer Networks and Communications (IJCNC)*, 2012.

[14] T. Mills and T. Mills, *Time Series Techniques for Economists*. Cambridge University Press, 1991.

[15] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, "Time series analysis: Forecasting and control. 3rd ed." *Englewood Cliffs, NJ: Prentice Hall*, 1994.

[16] N. C. Anand, C. Scoglio, and B. Natarajan, "Garch - non-linear time series model for traffic modeling and prediction," in *Proc. of IEEE Network Operations and Management Symposium (NOMS)*, 2008, pp. 694–697.

[17] R. Boutaba et al., "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 6, 2018.

[18] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying deep learning approaches for network traffic prediction," in *Proc. of International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 2353–2358.

[19] A. Azari, P. Papapetrou, S. Denic, and G. Peters, "Cellular traffic prediction and classification: a comparative evaluation of LSTM and ARIMA," 05 2019, https://arxiv.org/abs/1906.00939.

[20] M. Barabas, G. Boanea, A. B. Rus, V. Dobrota, and J. Domingo-Pascual, "Evaluation of network traffic prediction based on neural networks with multi-task learning and multiresolution decomposition," in *Proc. of IEEE Intelligent Computer Communication and Processing (ICCP)*, 2011, pp. 95–102.

[21] C. Guang, G. Jian, and D. Wei, "A time-series decomposed model of network traffic," in *Proc. of the First international conference on Advances in Natural Computation (ICNC)*, 2005, pp. 338–345.

[22] J. Xiao, Z. Xiao, D. Wang, J. Bai, V. Havyarimana, and F. Zeng, "Short-term traffic volume prediction by ensemble learning in concept drifting environments," *Knowledge-Based Systems*, vol. 164, pp. 213 – 225, 2019.

[23] K. Frans, J. Ho, X. Chen, P. Abbeel, and J. Schulman, "Meta learning shared hierarchies," in *Proc. of the 6th International Conference on Learning Representations (ICLR)*, 2018, pp. 1–11.

[24] G. Dán and N. Carlsson, "Dynamic content allocation for cloud-assisted service of periodic workloads," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, April 2014, pp. 853–861.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[26] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "The non-stochastic multiarmed bandit problem," *SIAM Journal on Computing*, vol. 32, no. 1, pp. 48–77, 2002.

[27] T. Han, Z. Zhang, M. Hu, G. Mao, X. Ge, Q. Li, and L. Wang, "Energy efficiency of cooperative base station sleep scheduling for vehicular networks," in *2014 IEEE 79th Vehicular Technology Conference (VTC Spring)*, May 2014, pp. 1–5.

Qing He (S'11–M'16) is Post-Doctoral Researcher with the School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Sweden. She received the B.Sc. and M.Sc. degrees in electrical engineering from Nanjing University, China, in 2001 and 2004, respectively, and her Ph.D. degree from the Department of Science and Technology, Linköping University, Sweden, in 2016. She has a second degree in Finance and had been working as a software designer and system engineer in Lucent Technologies Bell Labs until 2011. Her research interests include wireless network optimization, machine learning, and information theory.

Arash Moayyedi received his B.Sc. in computer engineering at Sharif University of Technology, Iran, in 2020. He was a guest researcher at the Division of Network and Systems Engineering, KTH Royal Institute of Technology, Sweden, in 2019, and a guest researcher at the Faculty of Computer Science of University of Vienna, Austria, in 2018. His research interest focuses on machine learning techniques in communication networks.

György Dán (M'07-SM'17) is Professor at KTH Royal Institute of Technology, Stockholm, Sweden. He received the M.Sc. in computer engineering from the Budapest University of Technology and Economics, Hungary in 1999, the M.Sc. in business administration from the Corvinus University of Budapest, Hungary in 2003, and the Ph.D. in Telecommunications from KTH in 2006. He worked as a consultant in the field of access networks, streaming media and videoconferencing 1999-2001. He was a visiting researcher at the Swedish Institute of Computer Science in 2008, a Fulbright research scholar at University of Illinois at Urbana-Champaign in 2012-2013, and an invited professor at EPFL in 2014-2015. He has been an area editor of Computer Communications since 2014 and of IEEE Trans. on Mobile Computing since 2019. His research interests include the design and analysis of content management and computing systems, game theoretical models of networked systems, and cyber-physical system security and resilience.

Georgios P. Koudouridis is a principal researcher at the Wireless System Lab. of Huawei Technologies Sweden AB, Kista, Sweden. He received a B.S. degree in Computer Sciences from the Dept. of Computer and Systems Sciences, Stockholm University (SU), Stockholm, Sweden, 1995, and a Ph.D. degree in electrical engineering on Telecommunications from Royal Institute of Technology (KTH), Stockholm, Sweden, 2016. Prior to joining Huawei in 2008, he was with Telia Research and TeliaSonera Mobile R&D. His research interests include radio resource management, spectrum allocation, self-organized networks and machine learning based optimisation in wireless networks.

Per Tengkvist (M'17) is senior System Architect at Huawei, with $>$ 20 years experience of R&D for GSM, UMTS, LTE and 5G. The main research interest is 5G Proactive RRM involving AI and Massive MIMO. Prior to working with Huawei, he was with Ericsson and Sectra.